Comparison of Constraint-Handling Techniques for Metaheuristic Optimization

Xing-Shi He[1], Qin-Wei Fan[1], Mehmet Karamanoglu[2], and Xin-She Yang[2⋆]

[1] College of Science, Xi'an Polytechnic University, Xi'an 710048, P. R. China.
[2] School of Science and Technology, Middlesex University, London, UK.

Abstract. Many design problems in engineering have highly nonlinear constraints and the proper handling of such constraints can be important to ensure solution quality. There are many different ways of handling constraints and different algorithms for optimization problems, which makes it difficult to choose for users. This paper compares six different constraint-handling techniques such as penalty methods, barrier functions, ϵ-constrained method, feasibility criteria and stochastic ranking. The pressure vessel design problem is solved by the flower pollination algorithm, and results show that stochastic ranking and ϵ-constrained method are most effective for this type of design optimization.

citation details: He X.S., Fan Q.W., Karamanoglu M., Yang X.S., (2019). Comparison of Constraint-Handling Techniques for Metaheuristic Optimization. In: Rodrigues J. et al. (eds) Computational Science ICCS 2019. ICCS 2019. Lecture Notes in Computer Science, vol 11538, pp. 357-366, Springer, Cham

https://doi.org/10.1007/978-3-030-22744-9_28

Keywords: Constraint-handling techniques · Feasibility · Flower pollination algorithm · Nature-inspired computation · Optimization.

1 Introduction

A vast majority of problems in engineering and science can be formulated as optimization problems with a set of inequality and equality constraints. However, such problems can be challenging to solve, not only because of the high nonlinearity of problem functions, but also because of the complex search domain shapes enclosed by various constraints. Consequently, both the choice of optimization algorithms and the ways of handling complex constraints are crucially important. Efficient algorithms may not exist for a given type of problem. Even with an efficient algorithm for a given problem, different ways of handling constraints may lead to varied accuracy. Thus, in addition to the comparison of

⋆ corresponding author

different algorithms, a systematic comparison of constraint-handling techniques is also needed [3,9,18,19,29].

There are many different algorithms for solving optimization problems [6,11]. One of the current trends is to use nature-inspired optimization algorithms to solve global optimization problems [28], and the algorithms such as genetic algorithm, differential evolution [24], particle swarm optimization [15], firefly algorithm and cuckoo search [28], and flower pollination algorithm [26] have demonstrated their flexibility and effectiveness. Thus, we will mainly use nature-inspired metaheuristic algorithms in this paper, and, more specifically, we will use the recent flower pollination algorithm (FPA) because its effectiveness and proved convergence [26,27,13]. In addition, even with any efficient algorithm, the constraints of optimization problems must be handled properly so that feasible solutions can be easily obtained. Otherwise, many solution attempts may be wasted and constraints may be violated [11,29]. There are many different constraint-handling techniques in the literature [8,12,16,17,23,32], and our focus will be on the comparison of different constraint-handling techniques for solving global optimization problems using metaheuristic algorithms.

Therefore, this paper is organized as follows. Section 2 provides a general formulation of optimization problems with a brief introduction to the flower pollination algorithm (FPA). Section 3 outlines different constraint-handling techniques. Section 4 uses FPA to solve a pressure vessel design problem with different ways of handling constraints where comparison of results will be presented. Finally, Section 5 concludes with some discussions.

2 Optimization

2.1 General Formulation

Though optimization problems can take many different forms in different applications, however, it can be formulated as a mathematical optimization problem in a D-dimensional design space as follows:

$$\text{minimize} \quad f(\boldsymbol{x}), \quad \boldsymbol{x} = (x_1, x_2, ..., x_D) \in \mathbb{R}^D, \tag{1}$$

subject to

$$\phi_i(\boldsymbol{x}) = 0, \ \ (i = 1, 2, ..., M), \tag{2}$$

$$\psi_j(\boldsymbol{x}) \leq 0, \ \ (j = 1, 2, ..., N), \tag{3}$$

where $\boldsymbol{x}$ is the vector of D design variables, and $\phi_i(\boldsymbol{x})$ and $\psi_j(\boldsymbol{x})$ are the equality constraints and inequality constraints, respectively. Classification of different optimization problems can be based on the problem functions. If these functions ($f(\boldsymbol{x})$, $\phi_i(\boldsymbol{x})$ and $\psi_j(\boldsymbol{x})$) are all linear, there are some efficient algorithms such as simplex methods. If problem functions are nonlinear, they may be more difficult to solve, though there are a wide range of techniques that can be reasonably effective [6,15,29]. However, global optimality may not be guaranteed in general.

2.2 Flower Pollination Algorithm

The flower pollination algorithm (FPA) as a population-based algorithm has been inspired by the characteristics of the pollination processes of flowering plants [26,27]. The main steps of the FPA have been designed to mimic some key characteristics of the pollination process, including biotic and abiotic pollination, flower constancy co-evolved between certain flower species and pollinators such as insects and animals, and the movement ranges of flower pollen of different flower species.

Briefly speaking, if $\boldsymbol{x}$ is the position vector that represents a solution in the design space to an optimization problem, this vector can be updated by

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \gamma L(\nu)(\boldsymbol{g}_* - \boldsymbol{x}_i^t), \tag{4}$$

which mimics the global step in the FPA. Here $\boldsymbol{g}_*$ is the best solution found so far in the whole population of n different candidate solutions, while γ is a scaling parameter, and $L(\nu)$ is a vector of random numbers, drawn from a Lévy distribution characterized by an exponent of ν.

Though the Lévy distribution is defined as

$$L(s) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} e^{-\frac{\gamma}{2(s-\mu)}} \frac{1}{(s-\mu)^{3/2}}, & (0 < \mu < s < +\infty), \\ 0, & \text{otherwise}, \end{cases} \tag{5}$$

which has an exponent of $3/2$, it can be generalised with an exponent of $1 \le \nu \le 2$ in the following form:

$$L(s,\nu) \sim \frac{A\nu\Gamma(\nu)\sin(\pi\nu/2)}{\pi|s|^{1+\nu}}, \tag{6}$$

where $s > 0$ is the step size, and A is a normalization constant. The Γ-function is given by

$$\Gamma(z) = \int_0^\infty u^{z-1} e^{-u} du. \tag{7}$$

In the special case when $z = k$ is an integer, it becomes $\Gamma(k) = (k-1)!$. The average distance d_L or search radius covered by Lévy flights takes the form

$$d_L^2 \sim t^{3-\nu}, \tag{8}$$

which increases typically faster than simple isotropic random walks such as Brownian motion because Lévy flights can have a few percent of moves with large steps in addition to many small steps [21].

The current solution $\boldsymbol{x}_i^t$ as a position vector can be modified locally by varying step sizes

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + U(\boldsymbol{x}_j^t - \boldsymbol{x}_k^t), \tag{9}$$

where U is a vector with each of its components being drawn from a uniform distribution. Loosely speaking, $\boldsymbol{x}_j^t$ and $\boldsymbol{x}_k^t$ can be considered as solutions representing pollen from different flower patches in different regions.

Due to the combination of local search and long-distance Lévy flights, FPA can usually have a higher capability for exploration. A recent theoretical analysis using Markov chain theory has confirmed that FPA can have guaranteed global convergence under the right conditions [13]. There are many variants of the flower pollination algorithm and a comprehensive review can be found in [2]. Due to its effectiveness, FPA has been applied to solve a wide range of optimization problems in real-world applications such as economic dispatch, EEG identification, and multiobjective optimization [1,22,27].

3 Constraint-Handling Techniques

There are many different constraint-handling techniques in the literature, ranging from traditional penalty methods and Lagrangian multipliers to more sophisticated adaptive methods and stochastic ranking [8,18,23]. In essence, penalty methods neatly transform a constrained optimization problem into a corresponding, unconstrained one by transforming its constraints in the revised objective, in terms of some additional penalty terms, and these penalty terms are usually functions of constraints. The advantage of this is that the optimization problem becomes unconstrained and thus the search domain has a regular shape without changing the locations of the optimality, but this modifies its original objective landscape, which may become less smooth. In addition, more parameters such as the penalty constants are introduced into the problem, and their values need to be set or tuned properly. In many cases, they can work surprisingly well if proper values are used, and the transformed unconstrained problem can be solved effectively by various optimization methods very accurately [11,29].

In this study, we aim to compare a few methods of handling constraints, and they are barrier functions, static penalty method, dynamic penalty method, feasibility method, ϵ-constrained method, and stochastic ranking.

3.1 Static Penalty and Dynamic Penalty Methods

Among various forms of the penalty method, the Powell-Skolnick approach [20] incorporates all the constraints with feasibility

$$\rho(\boldsymbol{x}) = \begin{cases} 1 + \mu\big[\sum_{j=1}^{N} \max\{0, \psi_j(\boldsymbol{x})\} + \sum_{i=1}^{M} |\phi_i(\boldsymbol{x})|\big], & \text{if not feasible,} \\ f(\boldsymbol{x}), & \text{if feasible,} \end{cases} \tag{10}$$

where the constant $\mu > 0$ is fixed, and thus this method is a static penalty method. This approach ranks the infeasible solution with a rank in the range from 1 to ∞, assuming the lower ranks correspond to better fitness for minimization problems.

In general, the penalty-based method transform the objective $f(\boldsymbol{x})$ into a modified objective Θ in the following form:

$$\Theta(\boldsymbol{x}) = f(\boldsymbol{x})[\text{objective}] + P(\boldsymbol{x})[\text{penalty}], \tag{11}$$

where the penalty term $P(\boldsymbol{x})$ can take different forms, depending on the actual ways or variants of constraint-handling methods. For example, a static penalty method uses

$$P(\boldsymbol{x}) = \sum_{i=1}^{M} \mu_i \phi_i^2(\boldsymbol{x}) + \sum_{j=1}^{N} \nu_j \max\{0, \psi_j(\boldsymbol{x})\}^2, \tag{12}$$

where $\mu_i > 0, \nu_j > 0$ are penalty constants or parameters. In order to avoid too many penalty parameters, a single penalty constant $\lambda > 0$ can be used, so that we have

$$P(\boldsymbol{x}) = \lambda \Big[\sum_{i=1}^{M} \phi_i^2(\boldsymbol{x}) + \sum_{j=1}^{N} \max\{0, \psi_j(\boldsymbol{x})\}^2 \Big]. \tag{13}$$

Since λ is fixed, independent of the iteration t, this basic form of penalty is the well-known static penalty method.

Studies show that it may be advantageous to vary λ during iterations [14,17], and the dynamic penalty method uses a gradually increasing λ in the following form [14]:

$$\lambda = (\alpha t)^{\beta}, \tag{14}$$

where $\alpha = 0.5$ and $\beta = 1, 2$ are used.

There are other forms of penalty functions. Recent studies suggested that adaptive penalty can be effective with varying penalty strength by considering the fitness of the solutions obtained during iterations [5,4,10].

3.2 Barrier Function Method

Though the equality constraints can be handled using Lagrangian multipliers, the inequalities need to be handled differently. One way is to use the barrier function [6], and the logarithmic barrier functions can be written as

$$L(\boldsymbol{x}) = -\mu \sum_{j=1}^{N} \log \Big[-\psi_j(\boldsymbol{x}) \Big], \tag{15}$$

where $\mu > 0$ can be varied during iterations (t). Here, we will use $\mu = 1/t$ in our implementations.

3.3 Feasibility Criteria

A feasibility-based constraint-handling technique, proposed by K. Deb [12], uses three feasible criteria as selection mechanisms: 1) the feasible solution is chosen first among one feasible solution and one infeasible solution; 2) the solution with a better (lower for minimization) objective value is preferred if two feasible solutions are compared; and 3) among two infeasible solutions, the one with the lower degree of constraint violation is preferred.

The degree of the violation of constraints can be approximately measured by the penalty term

$$P(\boldsymbol{x}) = \sum_{i=1}^{M} |\phi_i(\boldsymbol{x})| + \sum_{j=1}^{N} \max\{0, \psi_j(x)\}^2. \tag{16}$$

Such feasibility rules can loosely be considered as fitness ranking and preference of low constraint violation. Obviously, such feasibility rules can be absolute or relative, and thus can be extended to other forms [17].

3.4 Stochastic Ranking

Stochastic ranking (SR), developed by Runarsson and Yao in 2000 [23], is another constraint-handling technique, which becomes promising. In stochastic ranking, a control parameter $0 < p_f < 0$ is pre-defined by the user to balance feasibility and infeasibility, while no penalty parameter is used. The choice and preference between two solutions are mainly based on their relative objective values and the sum of constraint violations. Ranking of solutions can be done by any sorting algorithms such as the bubble sort.

The main step involves first to draw a uniformly-distributed u and compare with the pre-defined p_f. If $u < p_f$ or both solutions are feasible, then swap them if $f(\boldsymbol{x}_j) > f(\boldsymbol{x}_i)$. If both solutions are infeasible, swap if $P(\boldsymbol{x}_j) > P(\boldsymbol{x}_i)$. The aim is to select the minimum of the objective values and the lower degree of sum of the constraint violations.

The ranking is carried out according to the probability p_s

$$p_s = p_o p_f + p_v(1 - p_f), \tag{17}$$

where p_o is the probability of individual winning, based on its objective value, while p_v is the probability of winning of that individual solution, based on the violation of the constraints [23]. The probability of selection or winning among k comparison pairs among n solutions is based on a binomial distribution

$$p_w(k) = \frac{n!}{k!(n-k)!} p_s^k (1 - p_s)^{n-k}. \tag{18}$$

According to the value suggested by Runarsson and Yao [23], $p_f = 0.425$ will be used in this study.

3.5 The ϵ-Constrained Approach

Another technique for handling constraints, called the ϵ-constrained method, was developed by Takahama and Sakai [25], which consists of two steps: the relaxation limits for feasibility consideration and lexicographical ordering. Basically, two solutions $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ can be compared and ranked by their objective values $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$ and constraint violation ($P(\boldsymbol{x}_i)$ and $P(\boldsymbol{x}_j)$). That is

$$\{f(\boldsymbol{x}_i), P(\boldsymbol{x}_i)\} \leq \epsilon\{f(\boldsymbol{x}_j), P(\boldsymbol{x}_j)\}, \tag{19}$$

which is equivalent to the following conditions:

$$\begin{cases} f(\boldsymbol{x}_i) \le f(\boldsymbol{x}_j), & \text{if both } P(\boldsymbol{x}_i), P(\boldsymbol{x}_j) \le \epsilon \\ f(\boldsymbol{x}_i) \le f(\boldsymbol{x}_j), & \text{if } P(\boldsymbol{x}_i) = P(\boldsymbol{x}_j), \\ P(\boldsymbol{x}_i) \le P(\boldsymbol{x}_j), & \text{otherwise.} \end{cases} \tag{20}$$

Loosely speaking, the parameter $\epsilon \ge 0$ controls the level of comparison. In case of ϵ is very large, the comparison is mainly about objective values, while $\epsilon = 0$ corresponds to an ordering rule so that the objective minimization is preceded by lower or minimal degrees of the constraint violation [25,31].

4 Numerical Experiments and Comparison

In order to compare how these constraint-handling methods perform, we should use different case studies and different algorithms. However, due to the limit of space, here we only present the results for a design case study solved by the flower pollination algorithm. The optimal design of pressure vessels is a mixed integer programming, and it is a well-known benchmark in metaheuristic optimization for validating evolutionary algorithms.

4.1 Pressure Vessel Design

The pressure vessel design problem is a well-known benchmark that has been used by many researchers, and this problem is a mixed-type with four design variables. The overall design objective is to minimize the total cost of a cylindrical vessel, subject to some pre-defined volume and stress constraints. The four design variables are the thickness d_1 and d_2 for the head and body of the vessel, respectively, the inner radius r of the cylindrical section, and the length W of the cylindrical part [7,8]. The objective is to minimize the cost:

$$\text{minimize } f(\boldsymbol{x}) = 06224rWd_1 + 1.7781r^2d_2 + 19.64rd_1^2 + 3.1661Wd_1^2, \tag{21}$$

subject to four constraints:

$$g_1(\boldsymbol{x}) = -d_1 + 0.0193r \le 0, \quad g_2(\boldsymbol{x}) = -d_2 + 0.00954r \le 0, \tag{22}$$

$$g_3(\boldsymbol{x}) = -\frac{4\pi r^3}{3} - \pi r^2 W - 1296000 \le 0, \quad g_4(\boldsymbol{x}) = W - 240 \le 0. \tag{23}$$

The simple limits for the inner radius and length are: $10.0 \le r, W \le 200.0$.

However, due to some manufacturability requirements, it is necessary to set the thickness (d_1 and d_2) to be the integer multiples of a basic thickness of 0.0625 inches. That is

$$1 \times 0.0625 \le d_1, d_2 \le 99 \times 0.0625. \tag{24}$$

With four variables and four constraints, it seems not so hard to solve the problem. However, the first two variables are discrete, which makes the problem become a mixed integer programming problem. This benchmark has been studied

extensively by many researchers [7,28]. For many years, the true optimal solutions were not known due to the nonlinearity in its objective and constraints.

Now the true global optimal solution [30], based on the analytical analysis, is $f_{\min} = 6059.714335$ with $d_1 = 0.8125$, $d_2 = 0.4375$, $r = 40.098446$ and $W = 176.636596$. This allows us to compare the obtained solutions with the true solution in this study.

4.2 Comparison

Most penalty methods used in the literature require a high number of iterations, typically from 10 000 or 50 000 up to even 250 000 or 500 000 so as to get sufficient accurate results [8,12]. However, in order to see how these methods evolve throughout iterations, a much lower number of iterations will be used here. In fact, we will use $t_{\max} = 10000$, this allows us to see how errors will evolve over time for different methods. Other parameters are: $\lambda = 10^5$ for static penalty, $\alpha = 0.5$ and $\beta = 2$ for dynamic penalty. $\mu = 1/t$ is used for the barrier function, and $p_f = 0.425$ is used for stochastic ranking. In addition, $\epsilon = 1$ is used for the ϵ-constrained method. For the FPA, parameters are: population size $n = 40$, $p_a = 0.25$, $\gamma = 0.1$, and $\nu = 1.5$.

There are many different ways to compare simulation results, and the ranking results can largely depend on the performance measures used for comparison. Here, we will use the modified offline error E, similar to the error defined by Ameca-Alducin et al. [3]. We have $E = \frac{1}{N_{\max}} \sum_{t=1}^{N_{\max}} |f_{\min} - f_*^{(t)}|$ where $N_{\max}$ is the maximum number of iterations and we use $N_{\max} = 10000$. Here, $f_*^{(t)}$ is the best solution found by an algorithm during iteration t, and $f_{\min}$ is the known best solution from the literature, and it is the global minimum, based on analytical results for the pressure vessel design problem [30].

Table 1. Mean errors of the pressure vessel objective with 20 independent runs.

Method	Iteration ($t = 5000$)	Iteration $t = 10000$
Static Penalty	416.1	322.6
Dynamic Penalty	368.7	317.8
Barrier Function	497.9	421.3
Feasibility Approach	402.7	310.2
ϵ-Constrained	341.5	171.9
Stochastic Ranking	332.4	169.3

Six different constraint-handling methods are implemented in this study, and all methods can find the optimal solution $f_{\min} = 6059.714$ for $t_{\max} = 10000$. The results of 20 independent runs and the mean errors of the pressure vessel design objective values from the true optimal value are summarized in Table 1. As we can see, the errors are decreasing as iteration t becomes larger. Both stochastic

ranking and ϵ-constrained method obtained the best results, while the feasibility approach is very competitive. Barrier function approach seems to give the worse results. Both static penalty and dynamic penalty can work well, though dynamic penalty is better than static penalty.

5 Conclusions

This paper has compared six different constraint-handling techniques in the context of bio-inspired algorithms and nonlinear pressure vessel designs. The pressure vessel design problem is a nonlinear, mixed-integer programming problem and has been solved by using the FPA. The emphasis has been on the comparison of different ways of handling constraints. Our results have shown that both stochastic ranking and ϵ-constrained method obtained the best results.

Further studies will focus on the more extensive tests of different constraint-handling techniques and different algorithms over a wide range of benchmarks and design problems. More detailed parametric studies will also be carried out so as to gain insight into advantages and disadvantages as well as robustness of different constraint-handling techniques.

References

1. Abdelaziz, A.Y., Ali, E.S., Abd Elazim, S.M.: Combined economic and emission dispatch solution using flower pollination algorithm. Int. J. Electrical Power and Energy Systems **80**(2), 264-274 (2016)
2. Alyasseri, Z.A.A., Khader, A.T., Al-Betar, M.A., Awandallah, M.A., Yang, X.S.: Variants of the flower pollination algorithm: a review. In: Nature-Inspired Algorithms and Applied Optimization (Edited by Yang, X.S.), pp. 91-118. Springer, Cham (2018).
3. Ameca-Alducin, M.-Y., Hasani-Shoreh, M., Blaidie, W., Neumann, F., Mezura-Montes, E.: A comparison of constraint handling techniques for dynamica constrained optimization problems. arXiv:1802.05825 (2018). Last accessed 12 Feb 2019
4. Barbosa, H.J.C., Lemonge, A.C.C.: A new adaptive penalty scheme for genetic algorithms. Information Sciences **156**(5), 215-251 (2003)
5. Bean, J.C., Hadj-Alouane, A.B.: A dual genetic algorithm for bounded integer programs. Technical Report RT 92-52. Department of Industrial adn Operations Engineering, University of Michigan, Ann Arbor, USA (1992)
6. Boyd, S., Vandenberge, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
7. Cagnina, L.C., Esquivel, S.C., Coello, C.A.: Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica **32**(3), 319–326 (2008)
8. Coello, C.A.C.: Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry **41** (2), 113–127 (2000)
9. Coello, C.A.C.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering **191** (11-12), 1245-1287 (2002)

10. Coit, D.W., Smith, A.E., Tate, D.M.: Adaptive penalty methods for genetic optimization of constrained combinatorial problems. INFORMS Journal on Computing **6** (2), 173-182 (1996)
11. Deb, K. Optimization for Engineering Design: Algorithms and Examples. Prentice-Hall, New Delhi (1995)
12. Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering **186**(2-4), 311-338 (2000)
13. He, X.S., Yang, X.S., Karamanoglu, M., Zhao, Y.X.: Global convergence analysis of the flower pollination algorithm: a discrete-time Markov chain approach. Procedia Computer Science **108**(1), 1354-1363 (2017)
14. Joines, J., Houck, C.: On the use of non-stationary penalty functons to solve nonlinear constrained optimization problems with GAs. In: Proceedings of the First IEEE Conference on Evolutionary Computation (Edited by Fogel, D.), pp.579-584, IEEE Press, Orlando, Florida (1994)
15. Kennedy, J., Eberhart, E.C., Shi, Y.: Swarm Intelligence. Academic Press, London (2001)
16. Koziel, S., Michalewicz, Z.: Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. Evolutionary Computation **7**(1), 19–44 (1999)
17. Mezura-Montes, E.: Constraint-Handling in Evolutionary Optimization. Studies in Computational Intelligence, vol. 198, Springer, Berlin (2009)
18. Mezura-Montes, E., Coello, C.A.C.: Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm and Evolutionary Computation **1**(4), 173–194 (2011)
19. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation **4**(1), 1-32 (1996)
20. Powell, D., Skolnick, M.M.: Using genetic algorithms in engineering design optimization with non-linear constraints. In: Proceedings of the Fifth International Conference on Genetic Algorithms (Eds. Forrest, S.), pp. 424–431, Morgan Kaufmann, San Mateo, California (1993)
21. Pavlyukevich, I.: Lévy flights, non-local search and simulated annealing. J Computational Physics **226**(2), 1830-1844 (2007)
22. Rodrigues, D., Silva, G.F.A., Papa, J.P., Marana, A.N., Yang, X.S.: EEG-based person identificaiton through binary flower pollination algorithm. Expert Systems with Applications **62**(1), 81-90 (2016)
23. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Transactions on Evolutionary Computation **4**(3), 284–294 (2000)
24. Storn, R., Price, K.: Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim **11**(4), 341–59 (1997)
25. Takahama, T., Sakai, S.: Solving constrained optimization problems by the ϵ-constrained particle swarm optimizer with adaptive velocity limit control. In: Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems (CIS'2006). 7-9 June 2006. pp. 683-689, IEEE Publication, Bangkok (2006).
26. Yang, X.S.: Flower pollination algorithm for global optimization. In: Unconventional Computation and Natural Computation. Lecture Notes in Computer Science vol. 7445, 240–249, Springer, Berlin (2012)
27. Yang, X.S., Karamanoglu, M., He, X.S.: Flower pollination algorithm: a novel approach for multiobjective optimization. Engineering Optimization **46**(9), 1222-1237 (2014)
28. Yang, X.S.: Cuckoo Search and Firefly Algorithm: Theory and Applications. Studies in Computational Intelligence. vol. 516, Springer, Heidelberg (2014)

29. Yang, X.S.: Nature-Inspired Optimization Algorithms. Elsevier, London (2014)
30. Yang, X.S., Huyck, C., Karamanoglu, M., Khan, N.: True global optimality of the pressure vessel design problem: a benchmark for bio-inspired optimisation algorithms. Int. J. Bio-Inspired Computation **5**(6), 329-335 (2013)
31. Yang, X.S.: Optimization Technqiues and Applications with Examples. John Wiley and Sons, Hoboken, NJ (2018)
32. Yeniay, O.: Penalty function methods for constrained optimization with genetic algorithms. Mathematical and Computational Applications **10**(1), 45–56 (2005).